IN THE CLAIMS

1. (Currently Amended) A custom class loader apparatus configured to dynamically locate and load classes in a virtual machine in accordance with an associated dependency specification, the custom class loader comprising:

class loading logic configured to specifically and dynamically locate, define and load a class specified by name;

a list of peer class loaders arranged in accordance with the associated dependency specification and, list generation logic configured to generate said list when said specified class has been replaced or when said dependency specification has been modified;

a flag indicating whether said class has been replaced; and,

deference logic configured to defer said location, definition and loading of said specified class to said peer class loaders in said list.

2. (Currently Amended) The custom class loader apparatus of claim 1, wherein said flag comprises a dirty bit.

3. (Currently Amended) The custom class loader apparatus of claim 1, wherein said custom class loader conforms to the specification of a ~~JAVA(TM)~~ Java programming language version 1.2 delegation-style custom class loader.

4. (Originally Filed) In a custom class loader executing in a virtual machine, a method of coordinating class loading among cyclically dependent classes comprising:

receiving a request to load a specified class;

determining whether said specified class has been replaced;

if it is determined that said specified class has been replaced, constructing a new instance of the class loader and generating a list of peer class loaders to which location, definition and loading of said specified class are to be deferred in accordance with a dependency specification in the virtual machine; and,

deferring said location, definition and loading to said peer class loaders in said list.

5. (Originally Filed) The method of claim 4, wherein said determining step comprises checking a dirty bit in the class loader.

6. (Originally Filed) The method of claim 4, wherein said generating step comprises:

traversing each peer class loader in said dependency specification; and,

adding a reference for each said traversed peer class loader to said list.

7. (Originally Filed) The method of claim 4, wherein said dependency specification comprises a tree of nodes, each said node encapsulating a reference to a dependency of said specified class, one of said nodes encapsulating a reference to said specified class.

8. (Originally Filed) The method of claim 7, wherein said generating step comprises:

beginning with said one node encapsulating a reference to said specified class, traversing each node in said dependency specification using a depth-first traversal strategy until

encountering either a leaf node or a node encapsulating a reference to a dependency already referenced in said list;

responsive to said encountering, traversing each node in said dependency specification using a breadth-first traversal strategy until encountering said node encapsulating said reference to said specified class; and,

adding a reference for each traversed node to said list.

9. (Originally Filed) The method of claim 6, wherein said generating step further comprises adding at least one reference to a peer class loader to said list based upon a corresponding reference stored in a list of peer class loaders identified in one of said traversed peer class loaders.

10. (Originally Filed) The method of claim 5, further comprising setting said dirty bit responsive to said specified class being replaced.

11. (Originally Filed) The method of claim 10, further comprising setting each dirty bit in each peer class loader referenced in said list responsive to said specified class being replaced.

12. (Originally Filed) A machine readable storage having stored thereon a computer program for coordinating class loading among cyclically dependent classes in a custom class loader executing in a virtual machine, the computer program comprising a routine set of instructions for causing the machine to perform the steps of:

receiving a request to load a specified class;

determining whether said specified class has been replaced;

if it is determined that said specified class has been replaced, constructing a new instance of the class loader and generating a list of peer class loaders to which location, definition and loading of said specified class are to be deferred in accordance with a dependency specification in the virtual machine; and,

deferring said location, definition and loading to said peer class loaders in said list.

13. (Originally Filed) The machine readable storage of claim 12, wherein said determining step comprises checking a dirty bit in the class loader.

14. (Originally Filed) The machine readable storage of claim 12, wherein said generating step comprises:

traversing each peer class loader in said dependency specification; and,

adding a reference for each said traversed peer class loader to said list.

15. (Originally Filed) The machine readable storage of claim 12, wherein said dependency specification comprises a tree of nodes, each said node encapsulating a reference to a dependency of said specified class, one of said nodes encapsulating a reference to said specified class.

16. (Originally Filed) The machine readable storage of claim 15, wherein said generating step comprises:

beginning with said one node encapsulating a reference to said specified class, traversing each node in said dependency specification using a depth-first traversal strategy until encountering either a leaf node or a node encapsulating a reference to a dependency already referenced in said list;

responsive to said encountering, traversing each node in said dependency specification using a breadth-first traversal strategy until encountering said node encapsulating said reference to said specified class; and,

adding a reference for each traversed node to said list.

17. (Currently Amended) The machine readable storage of claim 12, wherein said generating step comprises:

traversing ~~each~~ at least one parent class loader associated with the class loader through to a primordial class loader; and,

adding a reference for each said traversed parent class loader to said list.

18. (Originally Filed) The machine readable storage of claim 17, wherein said generating step further comprises adding at least one reference to a parent class loader to said list based upon a corresponding reference stored in a list of parent class loaders identified in one of said traversed parent class loaders.

19. (Originally Filed) The machine readable storage of claim 13, further comprising setting said dirty bit responsive to said specified class being replaced.

20. (Originally Filed) The machine readable storage of claim 19, further comprising setting each dirty bit in each parent class loader referenced in said list responsive to said specified class being replaced.